```
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMM          MMM  TTTTTTTTTTTTTTT  HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT  LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM  MMM  MMM           TTT       HHH          HHH  RRR         RRR       TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRR         TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR   RRR           TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR      RRR        TTT         LLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
```

```
UU      UU  VV      VV  XX      XX   SSSSSSSS   QQQQQQ   RRRRRRRR   TTTTTTTTTT  RRRRRRRR     222222
UU      UU  VV      VV  XX      XX   SSSSSSSS   QQQQQQ   RRRRRRRR   TTTTTTTTTT  RRRRRRRR     222222
UU      UU  VV      VV  XX      XX  SS         QQ    QQ  RR      RR      TT     RR      RR  22    22
UU      UU  VV      VV  XX      XX  SS         QQ    QQ  RR      RR      TT     RR      RR  22    22
UU      UU  VV      VV    XX  XX    SS         QQ    QQ  RR      RR      TT     RR      RR        22
UU      UU  VV      VV    XX  XX    SS         QQ    QQ  RR      RR      TT     RR      RR        22
UU      UU  VV      VV      XX       SSSSSS    QQ    QQ  RRRRRRRR       TT     RRRRRRRR        22
UU      UU  VV      VV      XX       SSSSSS    QQ    QQ  RRRRRRRR       TT     RRRRRRRR      22
UU      UU  VV      VV    XX  XX           SS  QQ  QQ QQ  RR  RR         TT     RR  RR        22
UU      UU  VV      VV    XX  XX           SS  QQ  QQ QQ  RR  RR         TT     RR  RR        22
UU      UU    VV  VV     XX      XX         SS  QQ    QQ  RR    RR       TT     RR    RR    22
UU      UU    VV  VV     XX      XX         SS  QQ    QQ  RR    RR       TT     RR    RR    22          ....
UUUUUUUUUU      VV       XX      XX  SSSSSSSS     QQQQ QQ  RR      RR     TT     RR      RR  2222222222  ....
UUUUUUUUUU      VV       XX      XX  SSSSSSSS     QQQQ QQ  RR      RR     TT     RR      RR  2222222222  ....

LL              IIIIII     SSSSSSSS
LL              IIIIII     SSSSSSSS
LL                II     SS
LL                II     SS
LL                II     SS
LL                II     SS
LL                II       SSSSSS
LL                II       SSSSSS
LL                II             SS
LL                II             SS
LL                II             SS
LL                II             SS
LLLLLLLLLL      IIIIII     SSSSSSSS
LLLLLLLLLL      IIIIII     SSSSSSSS
```

Table of contents

(2) 52 HISTORY ; Detailed Current Edit History
(3) 84 DECLARATIONS ; Declarative Part of Module
(4) 122 MTH$SQRTR2 - Standard Single Precision Floating SQRT
(5) 197 MTH$SQRT_R2 - JSB SQRT routine

```
0000     1          .TITLE  UVX$SQRTR2      ; Floating Point Square Root routine
0000     2                                  ; (SQRT)
0000     3          .IDENT /1-016/          ; File: MTHSQRTR2.MAR   EDIT: JCW1016
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;
0000    29 ; FACILITY: MATH LIBRARY
0000    30 ;++
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ; MTH$SQRT_R2 is a special routine which is the same as MTH$SQRT except
0000    34 ; a faster non-standard JSB call is used with the argument in R0 and no
0000    35 ; registers are saved.
0000    36 ;
0000    37 ;--
0000    38 ;
0000    39 ; VERSION: 01
0000    40 ;
0000    41 ; HISTORY:
0000    42 ; AUTHOR:
0000    43 ;       Peter Yuo, 15-Oct-76: Version 01
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ; 01-1  Peter Yuo, 22-May-77
0000    48 ; 01-2 Peter Yuo, 31-May-77
0000    49 ; 01-16 Jeffrey C. Wiener 14-Jul-83
0000    50 ;
```

```
0000     52          .SBTTL  HISTORY ; Detailed Current Edit History
0000     53
0000     54
0000     55 ; ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE: none
0000     56 ;
0000     57 ; Edit History for Version 01 of MTH$SQRT
0000     58 ;
0000     59 ; 01-1  Code saving after code review
0000     60 ; 01-2  ROTL shift in garbage into highest bit. Use ASHL instead.
0000     61 ;       ADDL instruction after ADJUST has been changed into ADDW to prevent
0000     62 ;       overflow if R1<31:16> = FFFF and R0<31:16> = FFFF
0000     63 ; 01-3  Finish error handling 10-June-1977
0000     64 ; 01-5  MTH$$ERROR changed to MTH$$SIGNAL.
0000     65 ;       MTH$_... changed to MTH__....
0000     66 ;       Changed error handling mechanism. Put error result in R0 before
0000     67 ;       calling MTH$$SIGNAL in order to allow user modify error result.
0000     68 ; 01-6  Return -0.0 on negative arg.  TNH 20-Dec-77
0000     69 ; 01-7  Edit in Rich Lary's code bums.  JSB routine is now _R2.  JMT 19-Jan-78
0000     70 ; 01-9  Move .ENTRY symbol to module header.  TNH 14-Aug-78
0000     71 ; 1-010 - Put version number in standard format: three digit edit
0000     72 ;          numbers.  Also, update the copyright notice.  JBS 16-NOV-78
0000     73 ; 1-011 - Change MTH__SQUROONEG to MTH$K_SQUROONEG.  JBS 07-DEC-78
0000     74 ; 1-012 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     75 ; 1-013 - Declare externals.  SBL 17-May-1979
0000     76 ; 1-014 - Move MTH$SQRT_R2 to separate module.  JAW 26-Sep-1979.
0000     77 ; 1-015 - Change external references to G^.  RNH 06-Oct-81
0000     78 ; 1-016 - Changed the reference to a D_floating instruction to a G_floating
0000     79 ;          instruction. Also made other minor changes in the code to accomodate
0000     80 ;          this change. This was done to conform to the policy that G_floating,
0000     81 ;          not D_floating, should be used to backup F_floating point routines.
0000     82 ;          JCW 14-JUL-83.
```

```
                                        .SBTTL  DECLARATIONS    ; Declarative Part of Module

                                ;
                                ; INCLUDE FILES:
                                ;

                                ;
                                ; EXTERNAL SYMBOLS:
                                ;
                                        .DSABL  GBL
                                        .EXTRN  MTH$K_SQUROONEG
                                        .EXTRN  MTH$$SIGNAL

                                ;
                                ; EQUATED SYMBOLS:
                                ;
                                ; MACROS:       none
                                ;
                                ; PSECT DECLARATIONS:

                                        .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                                                                ; program section for math routines
                                ;
                                ; OWN STORAGE:  none
                                ;
                                ; CONSTANTS:

                                ;
                                ; Constants A and B chosen for k = odd
                                ;
13CD5FD4                        LF_ODD_A_E63    =       ^X13CD5FD4
3C4A2018                        LF_ODD_B_EM63   =       ^X3C4A2018
                                ;
                                ; Constants A and B chosen for k = even
                                ;
F61A4015                        LF_EVEN_A       =       ^XF61A4015
4B231FD7                        LF_EVEN_B_EM64  =       ^X4B231FD7
```

```
                     0000   122          .SBTTL  MTH$SQRTR2  - Standard Single Precision Floating SQRT
                     0000   123
                     0000   124
                     0000   125 ;++
                     0000   126 ; FUNCTIONAL DESCRIPTION:
                     0000   127 ;
                     0000   128 ; SQRT  - single precision floating point function
                     0000   129 ;
                     0000   130 ; SQRT(X) is computed using the following approximation technique:
                     0000   131 ;
                     0000   132 ;       If X <= 0 , error.  Let X = |X|.
                     0000   133 ;
                     0000   134 ;       Let X = 2**K * F where F is the fractional part.
                     0000   135 ;
                     0000   136 ;       If K = even, X = 2**(2P) * F,
                     0000   137 ;              SQRT(X) = 2**P * SQRT(F), 1/2 =< F < 1
                     0000   138 ;
                     0000   139 ;       If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
                     0000   140 ;              SQRT(X) = 2**(P+1) * SQRT(F/2), 1/4 =< F/2 < 1/2.
                     0000   141 ;
                     0000   142 ;       Let F' = A*F + B,
                     0000   143 ;                       A = 0.453730314(octal),
                     0000   144 ;                       B = 0.327226214(octal), for K = even.
                     0000   145 ;              = A*(F/2) + B,
                     0000   146 ;                       A = 0.650117146(octal),
                     0000   147 ;                       B = 0.230170444(octal), for K = odd.
                     0000   148 ;       and
                     0000   149 ;           K' = P,      for K = even
                     0000   150 ;              = P + 1   for K = odd.
                     0000   151 ;
                     0000   152 ;       Let Y0 = 2**K' * F' as a staight line approximation wthin the
                     0000   153 ;       given interval using coefficients A and B which minimize the
                     0000   154 ;       absolute error at the midpoint and endpoint.
                     0000   155 ;
                     0000   156 ;       Starting with Y0, two Newton-Raphson iterations are performed.
                     0000   157 ;
                     0000   158 ;       Y[n+1] = (1/2) * ( Y[n] + X/Y[n])
                     0000   159 ;
                     0000   160 ;       The relative error is < 10**-8.
                     0000   161 ;
                     0000   162 ; CALLING SEQUENCE:
                     0000   163 ;
                     0000   164 ;       sqrt.wf.v = MTH$SQRTR2(x.rf.r)
                     0000   165 ;
                     0000   166 ; INPUT PARAMETERS:
                     0000   167
                     0000   168          LONG = 4                        ; define longword multiplier
                     0000   169          x = 1 * LONG                    ; Contents of x is the argument
                     0000   170
                     0000   171 ; IMPLICIT INPUTS:      none
                     0000   172 ;
                     0000   173 ; OUTPUT PARAMETERS:
                     0000   174 ;
                     0000   175 ;       VALUE:  floating square root of the argument
                     0000   176 ;
                     0000   177 ; IMPLICIT OUTPUTS:     none
                     0000   178 ;
```

```
                          0000   179 ; COMPLETION CODES:     none
                          0000   180 ;
                          0000   181 ; SIDE EFFECTS:
                          0000   182 ;
                          0000   183 ; Signals: MTH$_SQUROONEG if X < 0.0 with reserved operand in R0 (copied to
                          0000   184 ; the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
                          0000   185 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
                          0000   186 ; operand -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1
                          0000   187 ;
                          0000   188 ; NOTE: This procedure disables floating point underflow, enables integer
                          0000   189 ; overflow, causes no floating overflow or other arithmetic traps, and
                          0000   190 ; preserves enables across the call.
                          0000   191 ;
                          0000   192 ;---
                          0000   193
                          0000   194
                          0000   195
```

```
                                0000   197          .SBTTL  MTH$SQRT_R2  - JSB SQRT routine
                                0000   198
                                0000   199 ; JSB SQRT - used by the standard, and directly.
                                0000   200 ;
                                0000   201 ; CALLING SEQUENCE:
                                0000   202 ;       save anything in R0:R2
                                0000   203 ;       MOVF    ...,R0                  ; input in R0
                                0000   204 ;       JSB     MTH$SQRT_R2
                                0000   205 ;       return with result in R0
                                0000   206 ;
                                0000   207 ; Note: This routine is written to avoid any integer overflows, floating overflows,
                                0000   208 ; floating underflows or divide by 0 conditions, whether enabled or not.
                                0000   209 ;
                                0000   210 ; REGISTERS USED:
                                0000   211 ;       R0 - floating argument then result
                                0000   212 ;       R1 - X saved for use during iteration
                                0000   213 ;       R2 - scratch
                                0000   214
                                0000   215 MTH$SQRT_R2::                        ; JSB routine for SQRT
                  51    50    50  0000   216          MOVF    R0, R1           ; test sign of X and save it in R1.
                        5D    15  0003   217          BLEQ    ZERO_NEG         ; branch to ZERO_NEG if X =< 0
                                0005   218 ;
                                0005   219 ; X > 0
                                0005   220 ;
                                0005   221 POS:
                                0005   222 ;       CLRL    -(SP)                   ; make room for 2nd half of
                                0005   223 ;                                       ; double length operand
                  52    50    3C  0005   224          MOVZWL  R0, R2           ; isolate low 16 bits (sign,exp,>fract) in R
                        52    94  0008   225          CLRB    R2               ; R2 now has sign and left 7 exp bits
                  50    52    AA  000A   226          BICW    R2, R0           ; clear sign and left 7 exp bits
                        50    95  000D   227          TSTB    R0               ; check low bit of exp
                        10    18  000F   228          BGEQ    EVEN             ; and branch if 1
        50    13CD5FD4    8F    44  0011   229          MULF    #LF_ODD_A_E63, R0       ; add 64 (half of bias) to (exponent-2)
                                0018   230                                       ; and start approximation calc
        50    3C4A2018    8F    40  0018   231          ADDF    #LF_ODD_B_EM63, R0      ; R0 = (first approx) * 2**-64
                        13    11  001F   232          BRB     ADJUST           ; go adjust
                                0021   233
                                0021   234 EVEN:
              50    2000    8F    A0  0021   235          ADDW    #^X2000, R0             ; exp is 0 - make it 64 (2**-64) for legalit
        50     F61A4015    8F    44  0026   236          MULF    #LF_EVEN_A, R0
        50     4B231FD7    8F    40  002D   237          ADDF    #LF_EVEN_B_EM64, R0      ; R0 = (first approx) * 2**-64
                                0034   238 ADJUST:
              52    52    1F    9C  0034   239          ROTL    #31, R2, R2              ; divide R2 (exp+bias) by 2,
                                0038   240                                       ; giving (exp/2+64)
                    50    52    A0  0038   241          ADDW    R2, R0           ; insert exp/2 in first approx and
                                003B   242                                       ; re-bias it.
                                003B   243
                                003B   244 ; first iteration - single precision is sufficient
                                003B   245 ;
              52    51    50    47  003B   246          DIVF3   R0, R1, R2       ; R2 = X/Y0
                    50    52    40  003F   247          ADDF    R2, R0           ; R0 = Y0 + X/Y0
              50    0080    8F    A2  0042   248          SUBW    #^X80, R0        ; R0 = Y1 = (1/2)(Y0 + X/Y0)
                                0047   249                                       ; no overflow possible
                                0047   250
                                0047   251 ; second iteration, do in double precision to get truncated( rather than
                                0047   252 ; rounded) result.
                                0047   253 ;
```

```
                                0047   254 ;;;     CLRL    R2                      ; lower part (X) = 0
                                0047   255 ;;;     DIVD    R0, R1                  ; divide Y1 into X with low-order
                                0047   256                                         ; 32 bits of Y1 garbage.  This doesn't
                                0047   257                                         ; effect accuracy, since Y1 innacurate
                                0047   258                                         ; anyway.
                                0047   259 ;       PUSHL   R1                      ; convert x and place on stack
                                0047   260 ;       CLRL    R1                      ; clear low part of Y1
                                0047   261 ;       DIVD3   R0, (SP)+, R1           ; divide Y1 into X
                                0047   262 ;       ADDF    R1, R0                  ; R0 = Y1 + higher_part(X/Y1)
                                0047   263
                      7E  51 99FD   0047   264         CVTFG   R1, -(SP)
                      50  50 99FD   004B   265         CVTFG   R0, R0
                   7E 8E  50 47FD   004F   266         DIVG3   R0, (SP)+, -(SP)
                      50  8E 40FD   0054   267         ADDG2   (SP)+, R0
                      50  50 33FD   0058   268         CVTGF   R0, R0
                                005C   269
                   50 0080 8F  A2   005C   270         SUBW    #^X80, R0               ; R0 = SQRT(X) = (1/2) (Y1 + X/Y1)
                               05   0061   271 SQRTX:  RSB                             ; return, R0 = result
                                0062   272
                                0062   273 ; X =< 0
                                0062   274 ;
                                0062   275 ZERO_NEG:
                          FD    13   0062   276         BEQL    SQRTX                   ; return with R0 = result = 0
                          6E    DD   0064   277         PUSHL   (SP)                    ; return PC from JSB routine
                    7E  00'8F   9A   0066   278         MOVZBL  #MTH$K_SQUROONEG, -(SP) ; condition value
                   50   01  0F   78   006A   279         ASHL    #15, #1, R0             ; R0 = result = reserved operand -0.0
                                006E   280                                         ; R0 goes to signal mechanism vector
                                006E   281                                         ; (CHF$L_MCH_R0/R1) so error handler
                                006E   282                                         ; can modify the result.
          00000000'GF   02  FB   006E   283         CALLS   #2, G^MTH$$SIGNAL       ; signal error and use real user's PC
                                0075   284                                         ; independent of CALL vs JSB
                               05   0075   285         RSB                             ; return - R0 restored from CHF$L_MCH_R0/R1
                                0076   286
                                0076   287         .END
```

```
ADJUST              00000034 R      01
EVEN                00000021 R      01
LF_EVEN_A       =   F61A4015
LF_EVEN_B_EM64  =   4B231FD7
LF_ODD_A_E63    =   13CD5FD4
LF_ODD_B_EM63   =   3C4A2018
LONG            =   00000004
MTH$$SIGNAL         ********   X    00
MTH$K_SQUROONEG     ********   X    00
MTH$SQRT_R2         00000000 RG     01
POS                 00000005 R      01
SQRTX               00000061 R      01
ZERO_NEG            00000062 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_MTH$CODE	00000076 (118.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.08	00:00:01.39
Command processing	116	00:00:00.64	00:00:04.70
Pass 1	79	00:00:00.79	00:00:03.50
Symbol table sort	0	00:00:00.00	00:00:00.14
Pass 2	64	00:00:00.70	00:00:02.31
Symbol table output	2	00:00:00.02	00:00:00.02
Psect synopsis output	3	00:00:00.02	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	297	00:00:02.25	00:00:12.16

The working set limit was 900 pages.
3796 bytes (8 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 14 non-local and 0 local symbols.
347 source lines were read in Pass 1, producing 8 object records in Pass 2.
1 page of virtual memory was used to define 1 macro.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:UVXSQRTR2/OBJ=OBJ$:UVXSQRTR2 MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MS
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NCP
MAP

UUXSQRTR2
LIS

UUXSQRT
LIS

NCPDEF
SDL

UUXSINCOS
LIS

NMADEF
SDL

NCP